An empirical investigation of multiple viewpoint reasoning in requirements engineering

Tim Menzies[1], Steve Easterbrook[1], Bashar Nuseibeh[2], Sam Waugh[3]

[1] NASA/WVU Software Research Lab, 100 University Drive, Fairmont, West Virginia, USA, <tim@menzies.com>; <easterbrook@ivv.nasa.gov>;
[2] Department of Computing, Imperial College, London, UK, <ban@doc.ic.ac.uk>
[3] Defence Science and Technology Organisation, Air Operations Division, Melbourne, Australia, <sam.waugh@dsto.defence.gov.au>

Abstract. Multiple viewpoints are often employed in Requirements Engineering to facilitate traceability to stakeholders, to structure the requirements process, and to provide richer modelling by incorporating multiple conflicting descriptions. For the latter goal, it is important to understand the benefits of delaying resolution of inconsistency, and hence whether the additional complexity of reasoning with inconsistent requirements models is worthwhile. This paper describes an empirical study of the utility of abductive reasoning over multiple worlds during domain modelling. In the study we used a range of different models (ranging from correct to very incorrect), different fanouts, different amounts of data available from the domain, and different modelling primitives for representing time. In the experiments there was no significant change in the expressive power of models that incorporate multiple conflicting viewpoints. Whilst this does not negate the advantages of viewpoints during requirements elicitation, it does suggest some limits to the utility of viewpoints during requirements modelling. The paper discusses the implications of this finding for requirements modelling and inconsistency management.

1 Introduction

Acquiring and consolidating software requirements from different stakeholders is a time-consuming and costly process. If this process is poorly managed, the specifications have to be repeatedly reworked or the runtime system has to be extensively modified. In viewpoint-based requirements engineering, an emphasis is placed on capturing separate descriptions of the viewpoints of different stakeholders, and on identifying and resolving conflicts between them (e.g. [8,12,23]). In their survey of viewpoints-based approaches, Darke & Shanks note that "If different perceptions of the same problem domain can exist, then it may not always be possible, or desirable, to develop a single integrated viewpoint [that] attempts to satisfy the needs of all stakeholders" [4]. In this paper we set out to test the extent to which it is necessary to maintain multiple conflicting viewpoints during requirements modelling.

Viewpoints have been widely used in requirements engineering for a number of different reasons. Primarily, the motivation has been the observation that different stakeholders will have different views and perceptions of the problem domain. However, viewpoints have also been used to characterize entities in a system's environment [17], to characterize different classes of users [29], to distinguish between stakeholder terminologies [31], and to partition the requirements process into loosely coupled workpieces [24].

A key advantage to the use of viewpoints is that inconsistencies between viewpoints can be tolerated [9]. Toleration of inconsistent viewpoints is beneficial for three different aspects of requirements engineering:

1. Stakeholder buy-in and traceability. By capturing separately different stakeholder viewpoints during elicitation, stakeholders can identify their contributions, and requirements information can be traced back to a source.
2. Structuring the process. By permitting parallel development of separate 'workpieces', with no hard constraint on consistency between them, the analysis and specification process can be distributed amongst a team of developers.
3. Structuring the descriptions. Richer requirements models can be obtained by separating out different concerns, employing multiple problem structures, and delaying resolution of conflicts.

However, toleration of inconsistency comes with a price. Reasoning about inconsistent requirements models is computationally expensive. Most existing requirements modelling and verification approaches assume a consistent model, and provide little or no support for managing inconsistencies. If inconsistency is to be tolerated during modelling and analysis, then multiple world reasoning is needed. Such reasoning must be able to identify inconsistencies, sort the model into consistent worlds, and compare and evaluate inferences from the alternative worlds. Computationally this is NP-hard[1]. Even so, practical systems can be built to do reasoning over inconsistent theories for reasonable sized problems. We have explored two general approaches for this, namely labelled paraconsistent logics [15] and graph-based abduction [21].

So, on the one hand, it is clear that multiple viewpoints play an important role in requirements elicitation. On the other hand, it is reasonable to assume that eventually one or more consistent specifications will be needed as the basis for design and implementation. At what point should we attempt to combine the multiple viewpoints into a single consistent model? In the past we have argued that the maintenance of inconsistent viewpoints during requirements modelling is important, as the inconsistencies indicate areas of uncertainty, where more stakeholder input is needed [10,12]. We have even argued that it is possible to leave some inconsistencies unresolved in baselined specifications, when the cost

[1] Take a set of propositional clauses, C and an algorithm A that generates the maximal consistent subsets of C. If A returns C then C is consistent (and therefore satisfiable). Hence A is a solution to SAT, an NP-complete problem. So A cannot be P-time computable.

of removal is greater than the risk of misinterpretation[2]. However, the trade-off between these advantages, and the computational complexity during analysis has not been investigated. In this paper, we describe an initial experiment in which we tested the utility of multiple worlds reasoning during requirements modelling.

The framework we use in this paper for exploring multiple worlds reasoning is graph-based abduction. Informally, abduction is the inference to the best explanation [25]. More precisely, abduction makes assumptions in order to complete some inference. Mutually exclusive assumptions are managed in separate worlds [22]. That is, given a theory containing contradictions, abduction sorts those contradictions into consistent portions. In this case, the theory is the union of the viewpoints of different stakeholders. Queries can be written to assess the different worlds. Abduction allows us to examine the trade-offs between different worlds.

The paper is structured as follows. We first briefly explain our abductive framework, together with the graphical notation we use, and show how this framework captures conflicting viewpoints during domain modelling. We then describe an experiment in which we mutated an initial model to obtain a range of conflicting viewpoints, and then measured the utility of the multiple viewpoints in explaining our dataset. In requirements terms, this is the equivalent to testing whether multiple viewpoints are needed in domain modelling in order to capture all of the desired behaviors. The experiment will show that, at least in the domain studied, different viewpoints are rare and that there is little benefit in using multiple world reasoning. We discuss the implications of this result for viewpoint-based requirements engineering, and propose some follow-up studies.

2 Abduction approach

This section offers an example of abductive-based domain modelling. Consider the model shown in Figure 1. This figure is written in the QCM language [22] by two economists: Dr. Thick and Dr. Thin. In QCM, variables have three states: *up*, *down* or *steady*. These values model the sign of the first derivative of these variables and model the rate of change in each value. Dependencies between them can be created as follows. The direct connection between *foreignSales* and *companyProfits* (denoted with plus signs) means that *companyProfits* being *up* or *down* should be connected back to *foreignSales* being *up* or *down* respectively. The inverse connection between *publicConfidence* and *inflation* (denoted with minus signs) means that *inflation* being *up* or *down* should be connected back to *publicConfidence* being *down* or *up* respectively. We assume that, somehow, we have knowledge of the relative costs of each inference step in the model:

[2] We observed a simple example in the Shuttle Flight Software Specifications, where an input variable was referred to as taking values *true* and *false* at one point, and *on* and *off* at another. Because the programming language to be used would accept either as synonyms, and the cost of correcting the specifications was large, the inconsistency was ignored.

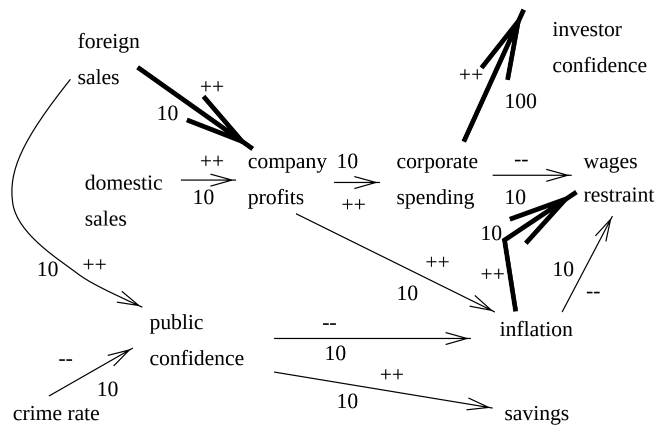

Fig. 1. A model from two experts.

each edge in the model is annotated with its numeric weight. Dr. Thick's and Dr. Thin's ideas are shown in thick and thin lines respectively. Note that our doctors disagree on the connection between *inflation* and *wagesRestraint*.

How can we test if Dr. Thick or Dr. Thin are saying anything sensible? One method is to use a library of known or desired behavior. Dr. Thick or Dr. Thin's ideas are sensible if they can reproduce that behavior. Further, one expert's model is better than others if that if that model can explain more known behavior than its competitors. However, selecting one of these expert viewpoints in preference to the other may not yield the best solution – it is unlikely that, for example, Dr. Thick is totally correct and Dr. Thin is totally wrong. It would be preferable to combine portions of Dr. Thick and Dr. Thin's viewpoints. More importantly, experience suggests that we should routinely expect domain models to reflect different and inconsistent viewpoints. In classical deductive logic, if we can prove a contradiction in a theory, then that theory becomes useless since anything at all can be inferred from a contradiction. Consider the case of *(foreignSales=up, domesticSales=down)* being inputs to the above economics model. We can now infer two contradictory conclusions: *companyProfits=up* and *companyProfits=down*. In classical deductive logic, we would have to declare our economics model useless. In this case, validating the models against our library of behaviors tells us nothing about the models.

2.1 Graph-based Abductive Validation

Graph-based abductive validation [20,22] allows us to perform inference on an inconsistent model, and hence check the relative claims of Dr. Thick and Dr. Thin. Graph-based abductive validation builds explanations (worlds) for each pair of inputs-outputs in the library of known or desired behaviors. Worlds are built by finding all possible proofs from outputs back to inputs across a directed graph like our economics model. Each maximally consistent subset of those proofs is a

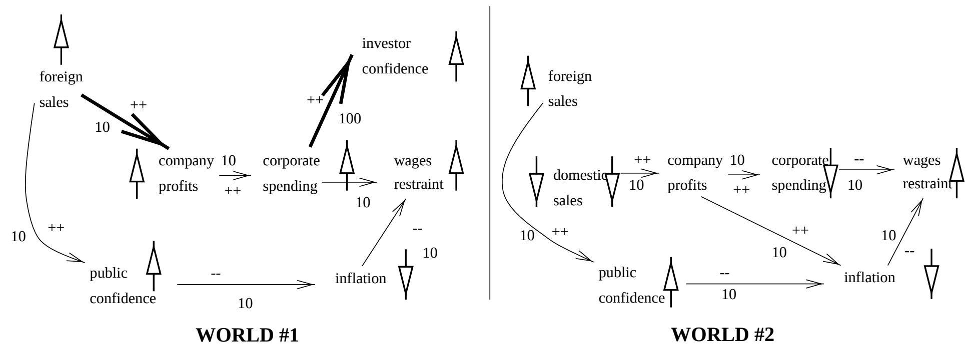

Fig. 2. Worlds from Figure 1.

world. Worlds are internally consistent. Contradictory assumptions are stored in separate worlds. Each world is scored via its intersection with the total number of outputs we are trying to explain. A model is then assessed by computing the largest score of its worlds.

This approach was first proposed by Feldman and Compton [11], then generalised and optimised by Menzies [20, 22]. Abductive validation has found a large number of previously unseen errors in scientific theories taken from international refereed publications. The errors had not previously been detected and had escaped peer review prior to publication.

To demonstrate how graph-based abductive validation allows us to validate a domain model containing multiple conflicting viewpoints, consider our economics model and the case where the inputs are *(foreignSales=up, domesticSales=down)* and the outputs are *(investorConfidence=up, inflation=down, wageRestraint=up)*. The six proofs `P` that can connect inputs to outputs are:

- `P.1`: *foreignSales=up, companyProfits=up, corporateSpending=up, investorConfidence=up.*
- `P.2`: *domesticSales=down, companyProfits=down, corporateSpending=down, wageRestraint=up.*
- `P.3`: *domesticSales=down, companyProfits=down, inflation=down.*
- `P.4`: *domesticSales=down, companyProfits=down, inflation=down, wagesRestraint=up.*
- `P.5`: *foreignSales=up, publicConfidence=up, inflation=down.*
- `P.6`: *foreignSales=up, publicConfidence=up, inflation=down, wageRestraint=up.*

Note that these proofs contain contradictory assumptions; e.g. *corporateSpending=up,* in `P.1` and *corporateSpending=down* in `P.2`. When we sort these proofs into maximal subsets that contain no contradictory assumptions, we arrive at the worlds shown in Figure 2. Note that world #1 covers all our output goals while world #2 only covers two-thirds of our outputs.

The use of viewpoints in requirements engineering is geared towards gaining stakeholder buy-in and facilitating discussion as much as it is about selecting the best model. Hence, this abductive approach does not offer automatic support for combining the ideas of different experts. However, it does support the automatic generation of reports describing the relative merits of the ideas of Dr. Thick and Dr. Thin as follows:

- We can explain all the behaviors in our dataset by combining portions of the viewpoints of Dr. Thick and Dr. Thin. (see world #1).
- We can find inconsistencies in the original viewpoints. For example, Dr. Thin's edges can be found in both worlds. Hence, with respect to the our dataset, (inputs *(foreignSales=up, domesticSales=down)* and outputs *(investorConfidence=up, inflation=down, wageRestraint=up)*), Dr Thin's original model is inconsistent.
- We can evaluate alternative explanations with respect to some cost function. In the example, Dr. Thin's ideas are cheaper than Dr. Thick. Consider the cost of world one which can support *investorConfidence=up*. This world contains the very expensive inference proposed by Dr. Thick. If we endorse only Dr. Thin, we get cheaper worlds but lose coverage of all outputs. Such a pragmatic trade off between cost and coverage could inform many debates during conflict resolution.

2.2 Advantages of this approach

This abductive approach has technical advantages over other approaches to conflict detection and resolution. Firstly, unlike existing viewpoints frameworks, it is not necessary for users to enter their requirements into explicitly labelled separate viewpoints, which are then assumed to be internally consistent. Recalling the above example, abduction can handle inconsistencies within the viewpoint of a single expert. Further, this approach can check if the explicitly labelled viewpoints are really different: if they don't generate different worlds when they are combined, then they are not truly different.

Secondly, this approach can find composite consistent models that use portions of each expert's knowledge to solve some task (see world #1, above).

Thirdly, graph-based abductive validation is not the JTMS-style [6] approach used in other conflict recognition and management systems (e.g. [28]). A JTMS searches for a single set of beliefs. Hence, by definition, a JTMS can only represent a single viewpoint at any one time. Our approach is more like the ATMS [5] than a JTMS. An ATMS maintains all consistent belief sets. We believe that an ATMS approach is better suited to conflict management in requirements engineering, since the different belief sets (viewpoints) are available for reflection.

Fourthly, one striking feature of other systems that support multiple-worlds (e.g. CAKE [28],TELOS [26]) is their implementation complexity. Rich and Feldman especially comment on the complexity of their heterogenous architecture [28]. We have found that it is easier to build efficient implementations [20,21] using the above graph-based approach than using purely logical approaches. These tools do not suffer from the restrictions of other tools. For example, while

Easterbrook's SYNOPTIC tool only permits comparisons of two viewpoints [7] (p113), our approach can compare N viewpoints.

Fifthly, the inference procedure described here avoids spurious state assignments. The state assignments proposed by a reasoner are its *envisionments*. Total envisionments are those behaviors which are possible, given some fixed collection of objects in some configuration. Extension generation in default logic [27] systems or the ATMS [5] produce total envisionments. A reasonable restriction on the total envisionments are the attainable envisionments; i.e. all behaviors possible from some given initial state. The QSIM qualitative reasoner uses attainable envisionments [18]. Graph-based abductive validation only finds the relevant envisionments; i.e. state assignments which can lead from inputs to outputs. Relevant envisionments answers the question: *Given some behavior of interest can these behaviors be reached given certain state assignments?* To answer this question with total or attainable envisionments, one must compute the total or attainable envisionments, then search them for the required behavior. This approach runs the risk of generating many behaviors that are irrelevant to the process of finding what percentage of known behaviors can be explained by a hypothetical model. For example, given the inputs and outputs of our above example, total envisionments would propose state assignments to *crimeRate* and attainable envisionments would propose state assignments to *savings*, even though these assignments are not relevant to reaching our output goals. To perform relevant envisionments, we restrict the search to the downstream transitive closure of the inputs and the upstream transitive closure of the outputs. For more details, see [22].

Lastly, the approach is simple enough that we can perform experiments on the utility of multiple world reasoning under different circumstances. The remainder of this paper describes such and experiment.

2.3 Limits to Abduction

There are important limitations to the type of abductive reasoning we have described. Selman and Levesque show that even when only one abductive explanation is required and the model is restricted to be acyclic, then abduction is NP-hard [30]. Bylander et.al. make a similar pessimistic conclusion [2].

The specific graph-based abduction validation procedure discussed above is also NP-hard. It grows proofs up from outputs back to inputs. As the proof grows, state assignments (e.g. *domesticSales=up*) are added to the proof. A proof must be consistent; i.e. it must not contain items that contradict other items in the proof. This proof invariant makes this procedure NP-hard. Gabow et.al. [13] showed that finding a directed path across a directed graph that has at most one of a set of forbidden pairs is NP-hard. Our forbidden pairs are assignments of different values to the same variable; e.g. the pairs *domesticSales=up* and *domesticSales=down*.

By reducing the problem of reasoning over multiple viewpoints to our abductive procedure, we have also shown that multiple viewpoint reasoning is NP-hard. This does not come as a surprise: the general problem of detecting inconsistencies in a specification is just a variant of the satisfiability problem. Pragmatic software engineers often build practical systems for problems that are theoretically NP-hard problems. Hence, merely showing that multiple viewpoint reasoning is NP-hard is not sufficient reason to abandon that approach. However, the experimental results discussed below are of more practical concern in questioning the need for multiple viewpoints reasoning.

3 Looking for Multiple Viewpoints

In exploring the utility of multiple viewpoints, we have found it useful to distinguish between use of multiple viewpoints during elicitation and their use during modelling and analysis. For the former, viewpoints can be used to represent different stakeholder's contributions, and to provide traceability back to an authority for each piece of information [14]. For the latter, viewpoints can be used to model and analyze conflicting information. In this paper, we are concerned primarily with the modelling and analysis issues, and in particular the need for multiple world reasoning. Viewpoints offer a number of other benefits for requirements modelling, including the use of multiple representation schemes, multiple problem structures, and the ability to partition the modelling process itself. However, if there is no inconsistency, then these benefits are essentially presentation issues: the same benefits could be achieved by taking projections and translations of a single, consistent model. That is not to say that such issues are trivial, but rather that it is the handling of inconsistency that makes viewpoints truly interesting for requirements modelling.

Our experiments are concerned with the use of viewpoints for requirements modelling and analysis. We would expect that if conflict and uncertainty in requirements is commonplace, one would find that multiple viewpoints would surface during modelling and analysis, regardless of whether they were used to structure the elicitation. As we have seen, our abductive framework provides a tool for identifying consistent worlds (i.e. viewpoints) in a model that contains inconsistencies, and indeed, multiple world reasoning is regarded as normal in abduction. Kakas et.al. [16] remark that a distinguishing feature of abduction is the generation of multiple explanations (worlds). Researchers into qualitative models (e.g. our economics model) often comment on the indeterminacy of such models (the generation of too many worlds). Clancy and Kuipers suggest that qualitative indeterminacy is the major restriction to the widespread adoption of qualitative reasoners [3].

Curiously, and contrary to the experience of Clancy, Kuipers, Kakas, et.al, graph-based abductive validation exhibits very little indeterminacy [19]. That is, when we checked for multiple worlds, we could not find them. This was such a surprising observation that we proceeded to conduct the following experiment. The aim of the experiment was to try and force graph-based abductive validation to generate numerous worlds. The experiment took an existing domain model, and mutated it to obtain a large number of alternative models, each of which was different from the original model.

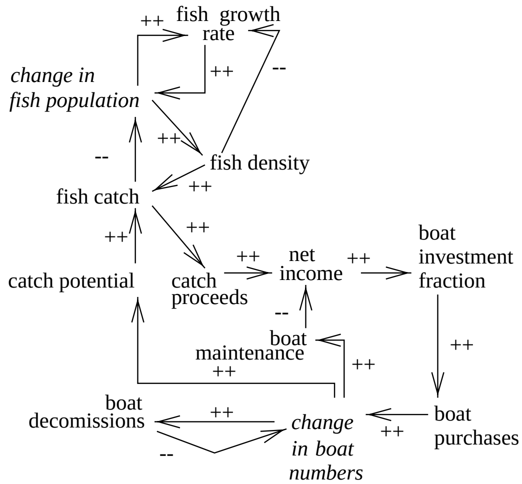

Fig. 3. The fisheries model. Adapted from [1] (pp135-141).

Firstly, some quantitative equations of a fisheries system were taken from Bossel [1] (pages 135-141) and converted into a QCM-style model, as shown in Figure 3. Note the two variables *change in boatNumbers* and *change in fishPopulation*. These change variables explicitly model the time rate of change of variables. The simulation data from the quantitative equations offered state assignments at every year. To handle such temporal simulations, the qualitative model was copied, once for every time tick in the simulation. That is, variables like *fishCatch* were copied to become *fishCatch@1*, *fishCatch@2*, etc. Variables at time *i* were connected to variables at time *i+1* using a *temporal linking policy* (discussed below).

Once we had a QCM model, we used graph-based abductive validation to try and reproduce data sets generated from the original quantitative equations. In our modelling exercise, this was essentially a validation step: does our model capture all the behaviors described in the original equations?

Then, to explore the multiple viewpoints issue, we built several *mutators* to generate 100,000s of different experimental treatments. The generated treatments contained (i) a range of different models (ranging from correct to very incorrect); (ii) models with different fanouts, (iii) different amounts of data available from the domain; (iv) different temporal linking policies.

One mutator added edges to the fisheries model. The original model has 12 nodes and 17 edges (fanout=17/12=1.4). This mutator added 0, 5, 10, 15, 20, 25 or 30 new edges at random (checking all the time that the added edges did not exist already in the model). That is, the model fanout was mutated from 1.4 to (17+30/12=3.9).

A second mutator corrupted the edges on the original fisheries model. This mutator selects N links at random in the fisheries model and flipped the annotation (*++* to - and visa versa). There are 17 edges in the fisheries model. Note

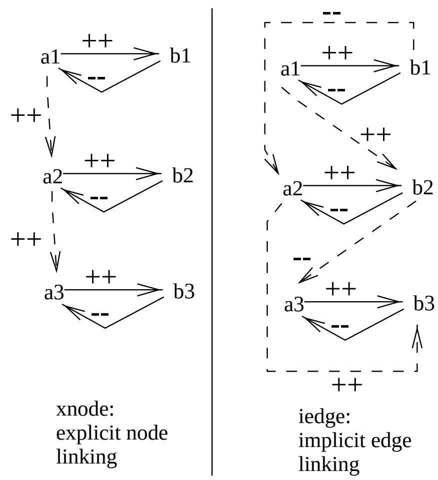

Fig. 4. *Direct(A,B)* and *inverse(B,A)* renamed over 3 time intervals using different time linking policies. Dashed lines indicate time traversal edges.

that as the number of edges mutated increases from 0 to 17, the mutated model becomes less and less like the original model. That is: at *mutations=0* we are processing the original (correct) fisheries model; at *mutations=17* we are processing a very incorrect fisheries model; at *mutations=2..16* we are processing progressively worse fisheries models.

A third mutator changed the amount of validation data available to our graph-based abduction. The complete set of Bossel equations provide values for all variables at all time points. The third mutator threw away some of that data to produce data sets with 0,10,..,90 percent of the variables unmeasured (denoted as U percent unmeasured).

A fourth mutator changed how the variables were connected across time. The XNODE temporal linking policy connects all the explictedly-marked temporal variables from time i to time $i+1$; e.g. *change in boatNumbers=up@1* to *change in boatNumbers=up@2*. Note that there are only two explicit temporal variables in fisheries. It was thought that, since the number of connections were so few, this could artificially restrict world generation. Hence, another time linking policy was defined which made many cross-time links. The IEDGE temporal linking policy took all edges from A to B in the fisheries model and connected $A@i$ to $B@i+1$. XNODE and IEDGE are compared in the following example. Consider a model with two variables, A and B, with a direct connection from A to B, and an inverse connection from B to A. Figure 4 shows how the XNODE and IEDGE linking policies expand this model over three time steps.

The above mutators were combined as follows. The Bossel equations were used to generate 105 pairs of inputs and outputs. For statistical validity, the following procedure was repeated 20 times for each of IEDGE and XNODE:

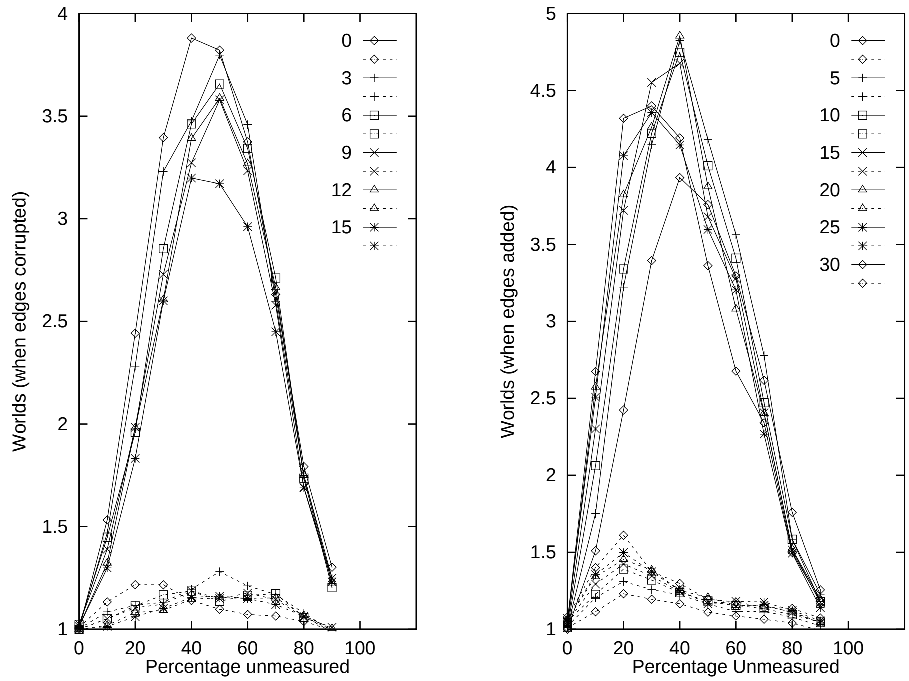

Fig. 5. IEDGE (solid lines), XNODE (dashed lines),

– 0 to 17 edges were corrupted, once for each value of U (0,10,..,90). This lead to 7200 models (20*2*10*18) executed over the 105 input-output pairs (7200*105= 756,000 runs).
– 0, 5, 10, 15, 20, 25 or 30 edges were added, once for each value of U leading to 20*2*10=400 models being executed 105 times (42,000 runs)

The results are shown in Figure 5.

Note the low number of worlds generated. Our reading of the literature (e.g. [3,16]) led us the expect far more worlds than those observed here (maximum=5) Also, note the *hump* shape in all the results graphs. As we decrease the amount of data available, there is less information available to constrain indeterminacy. Hence, initially, less data means more worlds. However, after some point (around 50 percent unmeasured), another effect dominates and the number of worlds decreases. We conjecture that relevant envisionments are the cause of the low number of worlds. World-generation is a function of the number of conflicting assumptions made by the reasoner. As the percentage of unmeasured variables increases, the size of the input and output sets decreases. In total envisionments, this has no effect on the number of assumptions made since total envisionments offers assumptions for all variables. However, attainable envisionments make fewer assumptions while relevant envisionments make even less. Hence, for low-

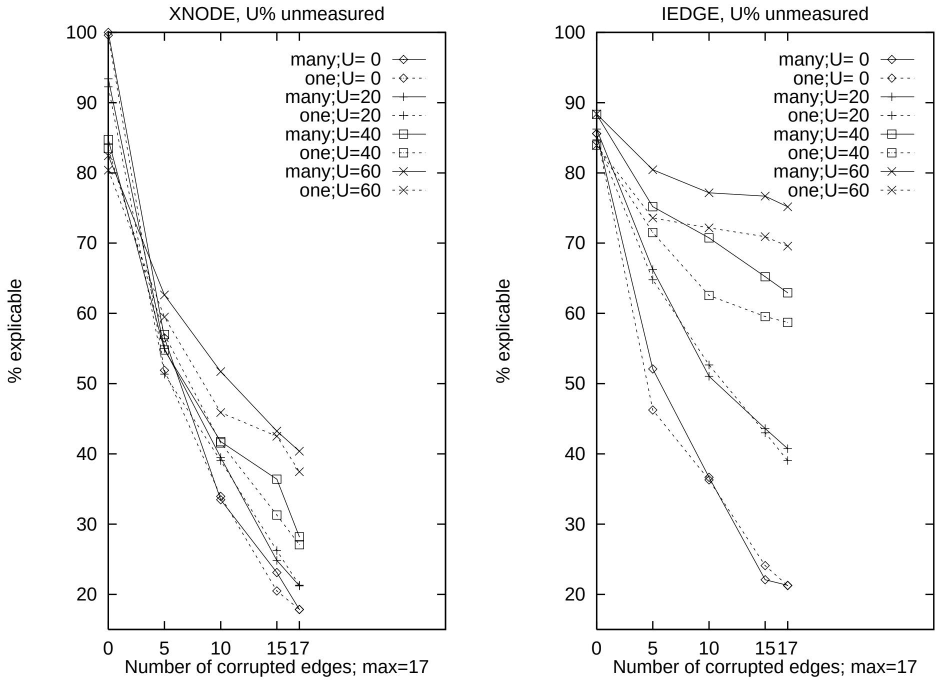

Fig. 6. Multiple-world abduction (solid line) vs one-world abduction (dashed line).

assumption envisionment policies (e.g. relevant envisionments), world-generation is reduced when the amount of data from the domain is reduced.

In summary, only certain interpretations of time (e.g. IEDGE) generate the multiple viewpoints that we expected. In other words, the generation of multiple worlds is extremely sensitive to the choice of modelling constructs, and some constructs will not generate multiple worlds. Our initial reaction was that if XNODE does not generate multiple worlds, then it is less expressive as a modelling language, and would be poorer at capturing all the behaviors in the original data. However, our next experiment contradicted this interpretation.

For the next experiment, we explored whether models that generated multiple worlds were more expressive. We modified the graph-based abductive validation procedure, so that instead of returning the world(s) that explained the most number of outputs, we returned any single world, chosen at random. The results of that one-world abduction run were compared to the results gained from full multiple-world abduction. For this experiment, we used the same test rig as was used in the edge corruption experiment described above; i.e. another 756,000 runs. A sample of those results are shown in Figure 6.

In these graphs, the percentage of behaviors found in the worlds is shown on the y-axis (labelled *percent explicable*). For multiple-world abduction, the maximum percentage is shown; i.e. this is the most explanations that the model can support. For one-world abduction, the percent of the one-world (chosen at random) is shown. Note that, at most, many-world reasoning was ten percent better than one-world reasoning (in the IEDGE graph for U=40 and 10 edges corrupted). The average improvement of many-world reasoning over one-world reasoning was 5.6 percent. That is, in millions of runs over thousands of models, there was very little difference seen in the worlds generated using one-world and multiple-world abduction.

4 Discussion

There are a number of conclusions we can draw from the experiments we have described. Multiple viewpoints reasoning is only useful if (i) the viewpoints are truly different and (ii) there is some value in incorporating multiple viewpoints in a requirements model. We have explored these two issues using our abductive framework. Abduction can check if some explicitly named viewpoints are truly different: if they don't generate different worlds when they are combined, then they are not truly different. Also, by comparing one-world abductive validation to multiple-world abductive validation, we can assess the merit of exploring multiple viewpoints. Experimentally, we have shown here that for a range of problems (different models ranging from correct to very incorrect, different fanouts, different amounts of data available from the domain, different temporal linking policies) multiple-world reasoning can only generate marginally better results than one-world reasoning (ten percent or less). Hence, in the domain explored by these experiments, there is no value in multiple viewpoints reasoning.

Before exploring the impact of this finding on requirements engineering, we need to consider the limitations of our experiment. The questions we need to address are whether the model we chose is representative of typical requirements models, whether the results scale, and whether there are important aspects to our model that we have ignored.

At first sight, our qualitative modelling language, QCM, may not seem appropriate in requirements engineering. However, our abductive approach merely performs abductive reasoning over a graph. Hence, we would expect the results to hold for any model that can be represented as a graph of similar shape to those generated in our experiments.

Secondly, our analysis is based on mutations of a single small model, *fisheries*. Perhaps an analysis of larger, more intricate models, would offer different conclusions? While we acknowledge this possibility, we note fisheries was just the initial model that seeded our mutators. Thousands of variants on fisheries were constructed, many of which were more complicated than fisheries (recall the first mutator added edges into the model). As to larger theories, we showed above that multiple viewpoint reasoning is NP-hard; i.e. this type of reasoning is will not be possible for very large models. Our approach shares this size restriction with all other techniques for reasoning over inconsistent models. In other words, for the range of models over which multiple world reasoning is feasible, it might not be useful.

Thirdly, our experiment assumed that it is possible and appropriate to assess the *worth* of a viewpoint along the lines of *what percent of known or desired behavior is found in that viewpoint?* This seems perfectly reasonable for requirements modelling, both for modelling an existing system and for modelling the desired behaviors of a new system. A problem here is that in requirements modelling, large datasets describing the desired behaviors may not be available *a priori*. In this case, the aim of requirements modelling is to generate the data from a model, and have the stakeholders validate the generated data. In this case our results still apply: we would not expect a multiple viewpoints model to generate many additional behaviors than a single viewpoint model. A second problem is that our approach ignores other measures of worth of a viewpoint. For example, an alternative measure of worth might be the degree to which including the viewpoint secures buy-in from a stakeholder. Our experiment does not consider such alternatives.

Fourthly, our scoring system for the worth of each viewpoint assumes there is a uniform distribution of goal *utilities*. That is, it measures worth by the number of behaviors explained, and ignores the fact that some behaviors may be more important than others. This is an incorrect assumption in most requirements processes. The differences in utilities of stakeholders' goals may be crucial when differences between their viewpoints are considered. We plan to conduct further experiments to test whether our results hold if the utilities on goals are varied.

So, within these limitations, our experiments appear to show that during requirements modelling, multiple viewpoint reasoning does not offer much advantage over single viewpoint reasoning. This does not mean that viewpoints-based requirements engineering should be abandoned. The experiments do not question the utility of viewpoints during elicitation. There are obvious benefits for capturing, separating and tracing the inputs of different stakeholders. This benefit almost certainly varies by domain, and there may be domains for which stakeholders have very little impact on software requirements[3]. Nevertheless, we expect that for most types of system, viewpoints offer a practical way of facilitating elicitation from multiple stakeholders.

In addition, there may well be domains where truly different viewpoints (of significantly different value) can be generated during requirements modelling. Also, note that in the second experiment, multiple world reasoning did improve the coverage of the desired behaviors by a few percent. In some domains, these few extra percent may be of vital importance to the application. For example, in a medical application, the few additional behaviors covered in the model might include those saving thousands of lives.

In other domains, the utility of multiple viewpoint reasoning will depend on the type of model built. Multiple-world reasoners are hard to build and understand. Our experiments indicate that there are some domains, or some parts of the requirements process where single world reasoning is sufficient.

[3] an example might be embedded software for device controllers, when the hardware design is already fixed.

Further experimentation is needed to confirm our findings, and to explore the limitations we have described above. For those domains in which the results hold we may wish to modify how viewpoints are applied. For example, if viewpoints are used for elicitation, our experiments would indicate that it is possible to combine the viewpoints into a single (consistent) requirements model earlier than we previously thought. We would expect that such a model may become inconsistent as it evolves. However, where inconsistencies do arise, we would not expect them to generate significantly different viewpoints, or if they do, these different viewpoints may not be worth considering. In the end, this result is not so surprising. It confirms what practitioners already know, namely that inconsistencies can be happily tolerated in specifications, because on the whole the inconsistencies may not matter much.

5 Acknowledgements

We would like to thank Iain Phillips and Francesca Toni of Imperial College for their helpful comments. This work was partially supported by NASA through cooperative agreement #NCC 2-979

References

1. H. Bossel. *Modeling and Simulations.* A.K. Peters Ltd, 1994. ISBN 1-56881-033-4.
2. T. Bylander, D. Allemang, M.C. M.C. Tanner, and J.R. Josephson. The Computational Complexity of Abduction. *Artificial Intelligence*, 49:25–60, 1991.
3. D.J. Clancy and B.K. Kuipers. Model decomposition and simulation: A component based qualitative simulation algorithm. In *AAAI-97*, 1997.
4. P. Darke and G. Shanks. Stakeholder viewpoints in requirements definition: A framework for understanding viewpoint development approaches. *Requirements Engineering*, 1(2):88–105, 1996.
5. J. DeKleer. An Assumption-Based TMS. *Artificial Intelligence*, 28:163–196, 1986.
6. J. Doyle. A truth maintenance system. *Artificial Intelligence*, 12:231–272, 1979.
7. S. Easterbrook. *Elicitation of Requirements from Multiple Perspectives.* PhD thesis, Imperial College of Science Technology and Medicine, University of London, 1991. Available from `http://research.ivv.nasa.gov/~steve/papers/index.html`.
8. S. Easterbrook. Handling conflicts between domain descriptions with computer-supported negotiation. *Knowledge Acquisition*, 3:255–289, 1991.
9. S. Easterbrook and B. Nuseibeh. Using viewpoints for inconsistency management. *BCS/IEE Software Engineering Journal*, pages 31–43, January 1996.
10. S. M. Easterbrook. Learning from inconsistency. In *Eighth International Workshop on Software Specification and Design (IWSSD-8)*, March 1996.
11. B. Feldman, P. Compton, and G. Smythe. Hypothesis Testing: an Appropriate Task for Knowledge-Based Systems. In *4th AAAI-Sponsored Knowledge Acquisition for Knowledge-based Systems Workshop Banff, Canada*, 1989.
12. A. Finkelstein, D. Gabbay, A. Hunter, J. Kramer, and B. Nuseibeh. Inconsistency handling in multi-perspective specification. *IEEE Transactions on Software Engineering*, 20(8):569–578, 1994.

13. H.N. Gabow, S.N. Maheshwari, and L. Osterweil. On two problems in the generation of program test paths. *IEEE Trans. Software Engrg*, SE-2:227–231, 1976.
14. O. Gotel and A. Finkelstein. Extended requirements traceability: Results of an industrial case study. In *International Symposium on Requirements Engineering (RE'97)*, pages 169–178, 1997.
15. A. Hunter and B. Nuseibeh. Managing inconsistent specifications: Reasoning, analysis and action. *ACM Transactions on Software Engineering and Methodology*, 7(4):00, 1998.
16. A.C. Kakas, R.A. Kowalski, and F. Toni. The role of abduction in logic programming. In C.J. Hogger D.M. Gabbay and J.A. Robinson, editors, *Handbook of Logic in Artificial Intelligence and Logic Programming 5*, pages 235–324. Oxford University Press, 1998.
17. G. Kotonya and I. Sommerville. Viewpoints for requirements definition. *IEE Software Engineering Journal*, 7:375–387, 1992.
18. B. Kuipers. Qualitative simulation. *Artificial Intelligence*, 29:229–338, 1986.
19. T.J. Menzies. *Principles for Generalised Testing of Knowledge Bases.* PhD thesis, University of New South Wales. Avaliable from `http://www.cse.unsw.edu.au/~timm/pub/docs/95thesis.ps.gz`, 1995.
20. T.J. Menzies. On the practicality of abductive validation. In *ECAI '96*, 1996. Available from `http://www.cse.unsw.edu.au/~timm/pub/docs/96abvalid.ps.gz`.
21. T.J. Menzies. Applications of abduction: Knowledge level modeling. *International Journal of Human Computer Studies*, 45:305–355, September, 1996. Available from `http://www.cse.unsw.edu.au/~timm/pub/docs/96abkl1.ps.gz`.
22. T.J. Menzies and P. Compton. Applications of abduction: Hypothesis testing of neuroendocrinological qualitative compartmental models. *Artificial Intelligence in Medicine*, 10:145–175, 1997. Available from `http://www.cse.unsw.edu.au/~timm/pub/docs/96aim.ps.gz`.
23. B. Nuseibeh. To be *and* not to be: On managing inconsistency in software development. In *Proceedings of 8th International Workshop on Software Specification and Design (IWSSD-8)*, pages 164–169. IEEE CS Press., 1997.
24. B. Nuseibeh, J. Kramer, and A. C. W. Finkelstein. A framework for expressing the relationships between multiple views in requirements specification. *IEEE Transactions on Software Engineering*, 20(10):760–773, 1994.
25. P. O'Rourke. Working notes of the 1990 spring symposium on automated abduction. Technical Report 90-32, University of California, Irvine, CA., 1990. September 27, 1990.
26. D. Plexousakis. Semantical and ontological considerations in telos: a language for knowledge representation. *Computational Intelligence*, 9(1), February 1993.
27. R. Reiter. A Logic for Default Reasoning. *Artificial Intelligence*, 13:81–132, 1980.
28. C. Rich and Y.A. Feldman. Seven layers of knowledge represeentation and reasoning in support of software development. *IEEE Transactions on Software Engineering*, 18(6):451–469, June 1992.
29. D. T. Ross. Applications and extensions of sadt. *IEEE Computer*, 18:25–34, 1985.
30. B. Selman and H.J. Levesque. Abductive and Default Reasoning: a Computational Core. In *AAAI '90*, pages 343–348, 1990.
31. R. Stamper. Social norms in requirements analysis: an outline of measur. In M. Jirotka and J. A. Goguen, editors, *Requirements Engineering: Social and Technical Issues*, pages 107–139. Academic Press, 1994.